REMARKS

Claims 1-27 are presented for prosecution. No claim is amended. No claim is cancelled.

Claims 1-6, 9-13, 15, 17-23, and 25-27 were rejected under 35 U.S.C. §103(a) as being unpatentable over Spyker et al. (U.S. Pat. 6,571,389), herein after referred to as Spyker, in view of Giroir et al. (U.S. Pat. 6,854,006), hereinafter referred to as Giroir.

Claims 7, 8, 14, 16, and 24 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Spyker in view of Giroir, as applied to claims 1-6, 9-13, 15, 17-23, and 25-27, and further in view of Schmidt et al. (U.S. Pat. 6,535,894), hereinafter referred to as Schmidt.

Applicants contacted Examiner Victor Lesniewski on October 26, 2005 and October 31, 2005 for clarification of the present claim rejections. Examiner Lesniewski explained that he is interpreting the claimed "port" in a manner consistent with the present application, and further elaborated upon the present claim rejections. In reference to the independent claims, as exemplified by claim 25, Examiner Lesniewski expressed the opinion that although the present invention, in practice, appears different from the cited prior art, he believes that Spyker may still read on the current claim language.

Specifically, Examiner Lesniewski explained that since Spyker describes a web browser in communication with the Internet (or other network using TCP/IP protocols), Spyker's web browser is inherently using a network port to convey said communication. Examiner Lesniewski conceded that his previous equating of Spyker, column 15, lines 12-20 to the recited operation of an application monitoring the network port for parameters was improper since Spyker, in column 15, lines 12-20, recites the use of local libraries already resident on the client machine and make no network access nor require any network ports. Examiner Lesniewski further explained that he understands that in the present invention, the claimed application, after being launched, monitors the network port used by the web browser to obtain and execute application parameters sent by the web browser to the network, but intended for the application.

Examiner Lesniewski further acknowledged that in col. 7, lines 66 to col. 68, line 3, Spyker states that his invention operates independently of the communication protocols used to send messages or files between the client and server. However, Examiner Lesniewski believed this text excerpt to be a self-contradictory statement since in the same excerpt Spyker goes on to state that he uses HTTP and TCP/IP (which are communication protocols) in an example when discussing "these message flows". Thus, if one were to ignore this Spyker excerpt, Examiner Lesniewski explained that he may then equate the transfer of "the dependencies for the application" (Spyker, col. 11, line 32 to col. 12, line 20, and especially col. 11, line 55-56) to the presently claimed "application parameters". In this case, Spyker may then be construed as describing a situation wherein his application is obtaining parameters through a network port.

In response, Applicants first intend to clarify Spyker's statement of col. 7, lines 66 to col. 68, line 3, and to explain that it is not a misstatement or a self-contradictory statement. Then, Applicants will explain why it is improper to equate Spyker's "dependencies for the application" with the presently claimed "application parameters", and furthermore, why even if such an equivalence is made, one still does not achieve the present invention.

The key to understanding Spyker's statement of col. 7, lines 66 to col. 68, line 3 is to identify what Spyker means when he refers to his "present invention" and to "these message flows". Spyker is making a distinction between his launched program and the mechanism by which he downloads and installs his program. That is, Spyker identifies his "present invention" as the installed program, and identifies "these message flows" as the communication between a web-browser and server to download and install the software modules necessary for installing his program. Although his program's multiple modules may be requested and downloaded through a network using a web-browser via HTTP on TCP/IP protocols, the actual launched program does not require any network communication, itself.

Basically, Spyker explains that his invention is a computer software program (col. 7, lines 53-54) that is JAVA-based, has the ease of use of an applet, but runs outside a web browser and thus has the robustness of a stand-alone application (col. 8, lines 15-18). As it is known, an applet is invoked from within a web browser, and Spyker explains how a web browser may be used to download all the code subroutines (i.e. modules or objects) for installing his program, select the appropriate JAVA run-time engine, and launch his program. To accomplish this, Spyker describes the use of a "dynamic loading software" tool (col. 7, lines 54-57) that is run by the web-browser to download (if necessary), install (if necessary), and launch his program (as is more fully explained below). It is important to note, however, that Spyker is differentiating the launched program (which he identifies as his "invention") from this dynamic loading software, which downloads and installs his program. Spyker repeats his emphasis that his launched program is independent of the web-browser in col. 8, lines 6-9.

With this in mind, the full excerpt of col. 7, line 53 to col. 8, line 3 can be more clearly understood, as follows:

> In the preferred embodiment, the <u>present invention is implemented as a computer software program</u>. Availability of a network connection is assumed, which must be operable at the time <u>when the dynamic loading software on a user's workstation is invoked</u>. In the preferred embodiment, <u>the present invention is implemented as one or more modules</u> (also referred to as code subroutines, or "objects" in object-oriented programming). The server to which the client computer connects may be functioning as a Web server, where that Web server provides services in response to requests from a client connected through the Internet. Alternatively, the server may be in a corporate intranet or extranet of which the client's workstation is a component. <u>The present invention operates independently of the communications protocol used to send messages or files between the client and server, although the HTTP (Hyper Text Transfer Protocol) protocol running on TCP/IP is used herein as an example when discussing these message flows</u>.

As can now be understood from the above excerpt, the invention is comprised of one or more modules, which are downloaded by Spyker's dynamic loading software, and the communication between the web browser and the web server instigated by the dynamic loading software is identified as "these message flows". Thus, Spyker is explicitly stating that his launched program operates independent of the message passed between the client and server, which is in direct conflict with the present invention. That is, the present invention requires that the launched program monitor messages between the client and server, and act upon any messages that include parameter changes intended for itself.

Now, Applicants address the meaning of Spyker's "dependencies for the application" as used in col. 11, line 55-56. Applicants refer first to col. 8, lines 4-40, which state,

> The present invention addresses a number of shortcomings in the Java environment, as will be discussed herein. The present invention also enables applets to be run without use of a browser, as if the applet was an application (and therefore all executable programs will be referred to hereinafter as "applications"). In this manner, the advantages of applets and the advantages of applications are combined. In particular, the disadvantages discussed earlier related to synchronizing applet code with the JVM level in a browser, different security APIs to invoke per browser, and the need to support multiple file archival formats are avoided by no longer using the browser as an execution environment. <u>The acronym Jobbi--which stands for "Java code Outside the Browser By IBM"--is used herein to refer to the implementation of the present invention. Each application in Jobbi has a **properties file** associated with it. The information in the properties file is used to describe the requirements of an application,</u> much as an applet tag would describe an applet's requirements in the current art. <u>Using this **properties information**, each application program can specify its **dependencies**</u>--<u>including the particular runtime environment that the application should run on--as well as the environment settings that are required for running the application.</u> <u>Multiple run-time environments (i.e. multiple versions of a JVM or JRE) can exist on a client machine,</u> where a single (shareable) copy of each run-time is accessible to those application programs which need it. A technique is defined herein for dynamically switching to the run-time environment which is required for a particular application, using information in the properties file. This is accomplished with little or no input from a human user. In this manner, older JVM levels are just as easily accessible for use at run-time as newer levels, freeing application developers from the need to update, recompile, and retest applications just to keep up with the moving target of the JVM level within the most recently released browser or operating system platform run-time environment.

From the above excerpt, it is clear that each Jobbi application (i.e. Spyker's program) includes a "properties file", and the properties file includes a listing of "dependencies". These "dependencies" name the type of operating environment the program depends upon for proper operation. In the above example, Spyker explains that some dependencies may specify the version of the Java Virtual Machine (JVM) and/or the Java Runtime Environment (JRE) it requires. The points are that: (1) Spyker's "dependencies" are bundled within a "properties file" downloaded <u>from a server to a client device</u> for program installation, and are <u>not</u> individual parameters passed <u>from a web browser to a server</u>; and (2) Spyker's "dependencies" are used for setting up the appropriate run-time environment within the client device in preparation for launching his program, whereas in the present invention the <u>already launched</u> program monitors the web browser's communication for individual parameter selections.

Since the appropriate run-time environment of an application is an integral part of a program's development (and not a parameter that may be selected by a user via a web-browser), Spyker explains in col. 8, line 51-55 that,

> "The properties definition process shown in FIG. 3 is a stand-alone process <u>performed by an application developer</u>, and is preferably <u>integrated into the normal application build process</u> which the developer uses during application development."

and further in col. 8, lines 63-67,

> "At Block 320, the <u>developer specifies values for the applicable properties of the application</u>, using his knowledge about the application's requirements. **These properties include application dependencies**, run-time requirements, etc., as will be described in more detail below with reference to FIG. 4."

With the understanding that the "dependencies" are defined by the software developer, and that the "dependencies" are packaged into a "properties" file used to define the run-time environment of Spyker's program prior to the program being launched, we can now turn to the Examiner's equating of "the dependencies for the application" (Spyker, col. 11, line 32 to col. 12, line 20, and especially col. 11, line 55-56) to the presently claimed "application parameters". Looking at the full process of Fig. 5, as explained in col. 11, lines 32 to col. 12, line 29, one can obtain a fuller understanding of Spyker's statement in col. 11, line 55-56.

Firstly, col. 11, lines 32 to 36 explain that Fig. 5a describes the process for locating the dependencies files on the server, and Fig. 5b describe the process for installing the downloaded files (please note that neither figure in this text excerpt describe the operation of Spyker's launched programmed, since the program has not yet been downloaded or installed). Col. 11, lines 44-52 state that the process begins by the web browser sending an HTTP download request to the server, which then locates the installation JAR files and transmits them to the web-server. Col. 11, lines 52-58 then explains how the downloaded JAR files are opened for program installation. Specifically, Spyker state,

> "...The processing of the Jobbi JAR file is explained in further detail in Blocks 535 and 540. <u>The properties information (see FIG. 4) is parsed at Block 535 to locate the dependencies</u> for the application requested at Block 505. When the dependency information is extracted, Block 540 checks to see whether the dependent item is installed."

Thus, the "dependencies" cited by the Examiner are not individual parameters sent from the web browser to the server, but rather are encased within a properties file, which in turn is compressed and packaged into an installation JAR file transmitted from the server. Again, Applicants respectfully emphasized that the JAR file is an installation file for installing Spyker's program, and thus Spyker's program cannot possibly be already launched nor monitoring any ports. Spyker explains that his installation process then determines if any of the listed dependencies (i.e. such as a specific version of the JAVA virtual machine needed for establishing the required run-time environment for the application) are not already resident on the client device. If any are not resident on the client device, then the needed dependencies are downloaded from the server in a recursive manner until all dependencies are made resident on the client device. If a needed dependency is not available on the server, then the server will locate the needed service on the web for download (col. 12, line 1-10). Specifically, Spyker states, col. 1, line 66 to col. 12, line 1, "This recursive invocation may be used to retrieve the run-time needed for the requested application, extensions needed for the application, etc."

Only after all the listed "dependencies" (i.e. runtime tools) that define the necessary run-time environment for Spyker's program have been downloaded onto the client device, will the installation of Spyker's program begin. Specifically, Spyker explains in col. 12, lines 24-29,

> "Once the <u>dependencies</u> have been fully processed as described with reference to Blocks 535 and 540, control returns to the mainline processing in Block 530, <u>where the Jobbi JAR file retrieved at 507 is installed into a Jobbi registry on the client machine</u>. This process is described in more detail in FIG. 5B."

During the installation process, all the required properties information (i.e. dependencies) needed to establish an appropriate run-time environment for Spyker's program are written into the Jobbi (i.e. software) registry (col. 12, line 30 to col. 14, line 41).

Description of the launching of Spyker's program begins in col. 14, line 42. In col. 14, line 57 to col. 15, line 10, Spyker explains that the launching process begins by readings the listed dependencies information written in the Jobbi registry, and starting the listed run-time environment tools. Only after the run-time environment is prepared, will the platform-dependent Java Native Interface (JNI) be invoked to launch the program using native libraries resident on the client device, and start execution of the program (see col. 15, lines 11-25).

As seen from the above, Spyker does not teach or suggest a program that, after being launched, monitors a network port for parameters sent by the web browser to the server, and executes any parameters it deems meant for it. Rather, Spyker teaches program that is downloaded through a network, but once downloaded and installed, will run independently of communication between the web browser and the server.

This Response After Final Rejection is believed clearly to place this application in condition for allowance and its entry is therefore believed proper under 37 C.F.R. §1.116. Accordingly, entry of this Response After Final Rejection, as an earnest attempt to advance prosecution and reduce the number of issues, is respectfully requested. Should the Examiner believe that issues remain outstanding, he/she is respectfully requested to contact applicants' undersigned attorney in an effort to resolve such issues and advance the case to issue.

For informational purposes, Applicants wish to also bring to the Examiner's attention U.S. Pat. 6,006,034 to Heath et al. cited in a Chinese search report of July 18, 2005 in a counterpart foreign application. U.S. Pat. 6,006,034 describes a network that checks the software version of an application on a network client before distributing software updates to the network client. Applicants believe that U.S. Pat. 6,006,034 is cumulative to information already of record in the present application and not material to patentability of the present invention in accordance with CFR §1.56 (b).

Applicants respectfully request reconsideration of the present application in view of the forgoing remarks.

Respectfully submitted,

[illegible]

Rosalio Haro
Registration No. 42,633

Please address all correspondence to:

Epson Research and Development, Inc.
Intellectual Property Department
150 River Oaks Parkway, Suite 225
San Jose, CA 95134
Customer No. 20178
Phone: (408) 952-6000
Facsimile: (408) 954-9058
Customer No. 20178

Date: November 3, 2005